Publicly Verifiable Private Aggregation of Time-Series Data

Bence Bakondi*, Andreas Peter†, Maarten Everts†‡, Pieter Hartel† and Willem Jonker*
*Database Group, University of Twente, the Netherlands
Email: {b.g.bakondi, willem.jonker}@utwente.nl
†Services, Cybersecurity and Safety Group, University of Twente, the Netherlands
Email: {a.peter, pieter.hartel}@utwente.nl
‡TNO, Netherlands Organisation for Applied Scientific Research
Email: maarten.everts@tno.nl

***Abstract*—Aggregation of time-series data offers the possibility to learn certain statistics over data periodically uploaded by different sources. In case of privacy sensitive data, it is desired to hide every data provider's individual values from the other participants (including the data aggregator). Existing privacy preserving time-series data aggregation schemes focus on the sum as aggregation means, since it is the most essential statistics used in many applications such as smart metering, participatory sensing, or appointment scheduling. However, all existing schemes have an important drawback: they do not provide verifiable outputs, thus users have to trust the data aggregator that it does not output fake values.**

We propose a *publicly verifiable* data aggregation scheme for privacy preserving time-series data summation. We prove its security and verifiability under the XDH assumption and a widely used, strong variant of the Co-CDH assumption. Moreover, our scheme offers low computation complexity on the users' side, which is essential in many applications.

I. Introduction

Data aggregation is the process of collecting information and expressing it in a summarized form, for instance for the realization of statistical analysis. It is relevant in many applications, such as smart metering (where, e.g., energy providers compute statistics on energy consumptions in an entire neighborhood), participatory sensing (where sensor readings from participating mobile devices are aggregated to form one large body of knowledge) and appointment scheduling (where availability information is aggregated to find a timeslot when everyone is available).

In these settings, privacy is often an important aspect. For instance, smart meter data from a single household can reveal the number of habitants and their working and sleeping habits; the information collected by the nodes of a participatory sensing network can be health or location related; and users of an appointment scheduler might not want to share their own, full time schedule. All these applications are based on time-series data, i.e., data is time-dependent and the aggregation is *session*-based: data providers periodically upload data and the aggregator computes statistics for each time period. For instance, when a smart meter records energy consumptions hourly, a session consists of records from the same neighborhood, in the same hour.

For such time-series data, Shi et al. [15] and Rastogi et al. [14] created the first privacy preserving data aggregation schemes of note. In [14], users send their data in an encrypted form to the data aggregator only if they are queried. Then, the data aggregator decrypts the sum of this data interacting with the users but without learning the data of *any* individual user. In [15], users send data on a session basis to a central server, which computes the sum of the submitted data in each session. One of the most important novelties they introduce is that the data aggregator decrypts the sum *without any further interaction* with any other participants (other than receiving the data itself). During the computation the server does not learn the individual values of the users and the computation is successful only if each user contributes. The need for every users' contribution is a key aspect of ensuring user privacy: when the statistics are based on the input of a small number of users, the individual values can be easily approximated, in particular if the standard deviation is known. The approach in [15] only works with a small plaintext space (although being sufficient in most applications, including those mentioned above). This restriction is eliminated by Joye and Libert [11], who provide a scheme to allow for a larger plaintext space. The price of their improvement is the loss of non-interactivity and the need for frequent updates of user secret keys. While the decryption itself is still done solely by the data aggregator without any further interaction, users update their keys interacting with a trusted authority. Leontiadis et al. [13] further improve the scheme. In their version the users can update their keys themselves without interaction with a trusted authority. This makes it possible to join or leave the scheme without significantly increasing the communication costs: the keys of different users are independent from each other, and the key of the data aggregator is computed in every session from the keys of the users present in that particular session. However, they still use a semi-trusted data collector and users have to interact with it while collecting data. Unfortunately, in most applications users are not always online, so interaction is an impractical requirement.

Besides the impractical user-interaction in [11], [13], and [14], *all* the above mentioned schemes for private aggregation of time-series data have an important drawback: they rely on

the trustworthiness of the server. An adversarial server can output a fake sum without anyone noticing. Even though user privacy is preserved in [15], [11], and [13], in most applications it is also essential that statistics revealed by the data aggregator be reliable. For instance, in case of health-related data, fake statistics can lead to incorrect medication or false research results, which are obviously very dangerous. One way to significantly improve reliability of a data aggregation scheme is to make the aggregation result verifiable, meaning that the aggregator outputs some "verification material" together with the actual aggregate, proving that the result is indeed the aggregate of the values sent by the users. In this way, undesirable changes in the data at the server's side (due to maliciousness or malfunctioning) can be noticed by any verifier.

In this work, we provide a *publicly verifiable* private aggregation scheme for time-series data. Just like [15], we focus on the most relevant statistics, namely the *sum* and we work in the same non-interactive setting where a central server collects encrypted data from users on a session basis, and it is only able to decrypt the sum of the data in each session. Decryption is possible only if every user contributes its data and the scheme is non-interactive (except for the data upload).

Related Work. In a private time-series data aggregation scheme, data providers send data periodically to a central data aggregator, which can only decrypt aggregated statistics (and nothing else) *without any further interaction* with the users [15]. Each data provider sends data only once per period and the data aggregator computes some statistics over data from the same period, without learning anything about any data provider's individual data.

Besides the literature mentioned above, there are other related schemes on privacy preserving data aggregation in various application settings: Ács et al. [1], Chan et al. [8] and Kursawe et al. [12] provide schemes for usage in the context of smart metering. Furthermore, De Cristofaro and Soriente [9] and Shi et al. [16] give a constructions for participatory sensing applications and Bilogrevic et al. [4] and Bilogrevic et al. [3] offer appointment scheduling schemes, but *none* of them can deal with verifiability. Moreover, in [9] and [16] the decryption is done separately from the data aggregation (just as in [4] and [3]) and there are intermediary actors between the data-providers and the data aggregator, which is often rather impractical.

Outline of the paper. In Section II we define what a Publicly Verifiable Private Data Aggregation (PV-PDA) scheme is and in Section II-A we introduce a general security and verifiability definition for PV-PDA schemes. In Section II-B we present our scheme and in Section II-C we prove its security and verifiability according to our security definition. In Section III we analyse the computational costs of our scheme and Section IV concludes the paper with a summary and a list of open challenges.

Notation and preliminaries. We indicate vectors with a bold font, the vector space (over $\mathbb{F}_p$) generated by $\left(\mathbf{v}^{(1)} = (v_1^{(1)}, \dots v_m^{(1)}), \dots, \mathbf{v}^{(n)}\right)$, for $n, m \in \mathbb{N}$ is denoted by $\langle \mathbf{v}^{(1)}, \dots, \mathbf{v}^{(n)} \rangle$. The orthogonal complement of vector subspace W is $W^{\perp}$ and $\mathbf{vw}$ denotes the inner product of $\mathbf{v}$ and $\mathbf{w}$.

In the following, we will apply two widely used assumptions: the *External Diffie-Hellman (XDH) assumption* and a strong variant of the *Computational Co-Diffie-Hellman (Co-CDH) assumption* which we will both explain momentarily.

The XDH assumption [2] says that there exist two distinct groups $\mathbb{G}_1, \mathbb{G}_2$ such that there exists an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and the following four problems are hard: the Discrete Logarithm Problem in both $\mathbb{G}_1$ and $\mathbb{G}_2$ (finding x from $g, g^x \in \mathbb{G}$), the Computational Diffie-Hellman problem in both $\mathbb{G}_1$ and $\mathbb{G}_2$ (computing g^{xy} from $g, g^x, g^y \in \mathbb{G}$), the Co-Computational Diffie-Hellman problem (Co-CDH, finding $g^x \in \mathbb{G}_1$ when $g \in \mathbb{G}_1$ and $h, h^x \in \mathbb{G}_2$ are known) and the Decisional Diffie-Hellman Problem in $\mathbb{G}_1$ (DDH, given $z, g^x, g^y \in \mathbb{G}$, deciding if $z = g^{xy}$ or not).

Recall that a DDH instance is a $(\mathbb{G} = \langle g \rangle, g, g^x, g^y, z)$ tuple, where $|\mathbb{G}| = p$, $x, y \in_R \mathbb{Z}_p$ and $z \in \mathbb{G}$. In the DDH problem the challenger flips a random bit and generates a DDH instance, where $z = g^{xy}$ if the random bit is 1, and $z \in_R \mathbb{G}$ if the random bit is 0. The adversary outputs a bit b' and wins if $b' = b$. The advantage of the adversary is $|\frac{1}{2} - \Pr[b' = b]|$.

To prove verifiability of the scheme, we also use the strong Co-CDH problem, introduced by Boneh et al. [7]. It says that in a *type III bilinear group tuple* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ [10] (i.e. $\mathbb{G}_1 \neq \mathbb{G}_2$ and there are no efficiently computable isomorphisms $\mathbb{G}_1 \to \mathbb{G}_2$ or $\mathbb{G}_2 \to \mathbb{G}_1$) no polynomial time algorithm can compute $g^{\alpha} \in \mathbb{G}_1$ given $h, h^{\alpha} \in \mathbb{G}_2$ and $f, f^{\alpha}, g \in \mathbb{G}_1$. A game for the strong variant of Co-CDH is the following. The challenger generates a type III bilinear group tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, and chooses $\alpha \in_R \mathbb{Z}_p$, $h \in_R \mathbb{G}_2$, $f, g \in_R \mathbb{G}_1$ and sends $h, h^{\alpha}, f, f^{\alpha}, g$ to the adversary. The adversary is supposed to output $g' \in \mathbb{G}_1$ and wins the game if $g' = g^{\alpha}$. The advantage of the adversary is $|\Pr[g' = g^{\alpha}] - \frac{1}{|\mathbb{G}|}|$.

II. Publicly Verifiable Private Data Aggregation

A Publicly Verifiable Private Data Aggregation scheme (PV-PDA) has two classes of actors: data providers (we refer to them as *users*), who upload privacy sensitive data to a central *Service Provider* (*SP*) who collects and aggregates the data from the users without being able to learn at any point, anything else about the original values other than the aggregation itself. We focus on the sum of the users' data as the aggregate function.

In a PV-PDA scheme, users send data to the SP on a session basis. Sessions are independent from each other, users can send data only once per session, and messages from different sessions cannot be aggregated together. At the end of each session, the SP outputs the *publicly verifiable* sum of the values from that particular session if, and only if *every* user contributes his data.

The following definition formalizes the described functionality (for a conceptual visualization, see Figure 1).

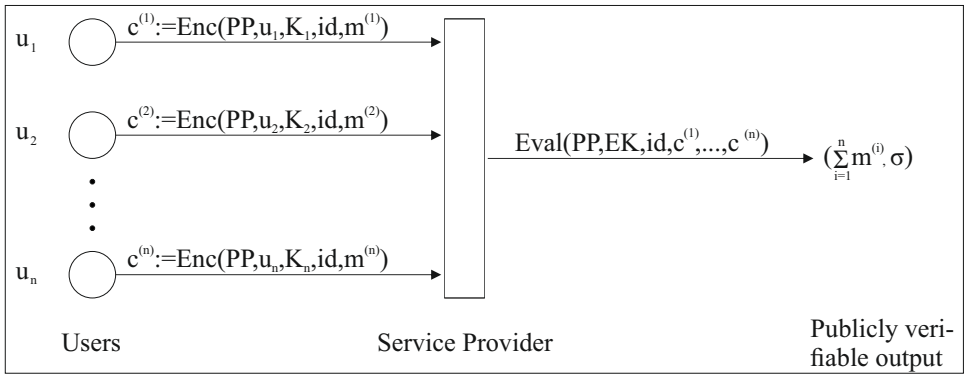

Figure 1: **Structure of the PV-PDA scheme.** In every session, users send their data encrypted to the Service Provider who outputs the sum of the data together with a verification material σ (if and only if every user contributed).

Definition 1 (Publicly Verifiable Private Data Aggregation): A Publicly Verifiable Private Data Aggregation is a tuple of the following four polynomial-time algorithms:

- $(PP, EK, K_1, \ldots, K_n) \leftarrow \texttt{Setup}(\lambda, n)$: given a security parameter λ and a number of users n as input, this algorithm outputs public parameters PP, an evaluation key EK and n user secret keys $K_1, \ldots, K_n$.
- $c^{(i)} \leftarrow \texttt{Enc}(PP, u_i, K_i, id, \mathbf{m}^{(i)})$: given public parameters PP, a user identifier $u_i \in [1, n]$, u_i's secret key K_i, a session identifier id and a message $\mathbf{m}^{(i)}$, it outputs a ciphertext $c^{(i)}$.
- $(\mathbf{M}, \sigma) \leftarrow \texttt{Eval}(PP, EK, id, c^{(1)}, \ldots, c^{(n)})$: given the public parameters PP, the evaluation key EK, the ciphertexts $c^{(1)}, \ldots, c^{(n)}$, and the session identifier id, it outputs $\mathbf{M} = \sum_{i=1}^{n} \mathbf{m}^{(i)}$ and verification material σ.
- $b \leftarrow \texttt{Verify}(PP, id, \mathbf{M}, \sigma)$: given the public parameters PP, message $\mathbf{M}$, verification material σ, and session identifier id, it outputs a bit b. $b = 1$ means 'acceptance', while $b = 0$ means 'rejection'.

We require that for any $(PP, EK, K_1, \ldots, K_n)$ obtained from the Setup algorithm and for any id, the output of $\texttt{Verify}(PP, id, \mathbf{M}, \sigma)$ is 1 if and only if $(\mathbf{M}, \sigma) \leftarrow \texttt{Eval}(PP, EK, id, \texttt{Enc}(PP, u_1, K_1, id, \mathbf{m}^{(1)}), \ldots, \texttt{Enc}(PP, u_n, K_n, id, \mathbf{m}^{(n)}))$. We emphasize that the verification material obtained from the Eval algorithm is generated without using any of the user secret keys.

A. Security and Verifiability

With regards to the *security* of a PV-PDA scheme, the intuition is that while the SP is still able to learn the sum for every session (as long as it has each user's contribution), it cannot learn anything about the individual values. To make this a little more precise: (1) messages from any user in any session are computationally indistinguishable, (2) messages from different sessions are uncombinable (the SP cannot compute their sum), and (3) it is impossible to compute the sum of the messages sent to the SP if not every user contributes its data in that session. Regarding *verifiability*, the SP should be able to prove that the sum he outputs is indeed the sum of the users' values in that session. We leave collusion attacks aside. The above described intuition is formalized in Definition 2 through a game-based security model.

Definition 2: A PV-PDA (Publicly Verifiable Private Data Aggregation) scheme is *secure* and *verifiable* if there is no PPT (probabilistic polynomial time) adversary $\mathcal{A}$ that has a non-negligible advantage in the security parameter λ against Game 1 or Game 2.

Game 1 (Security)

Setup. The challenger obtains $(PP, EK, \{K_i\}_{i=1}^{n}) \leftarrow \texttt{Setup}(\lambda, n)$ and gives EK and PP to the adversary $\mathcal{A}$.

Query 1. $\mathcal{A}$ adaptively chooses a sequence of identifiers and a sequence of n-tuples of messages, and sends them to the challenger. The challenger encrypts the first tuple of messages under the first identifier, the second tuple of messages under the second identifier, etc. and sends these encryptions to the adversary $\mathcal{A}$. In each tuple, the i-th message is encrypted with u_i's secret key for $i = 1, \ldots, n$.

Challenge. $\mathcal{A}$ chooses a session identifier $id \in \{0,1\}^{\lambda}$ different from any identifier used in the first query phase, and two n-tuples of messages with the same sum and sends these to the challenger. The two tuples are $\mathbf{m}_0 = \left(\mathbf{m}_0^{(1)}, \ldots, \mathbf{m}_0^{(n)}\right)$ and $\mathbf{m}_1 = \left(\mathbf{m}_1^{(1)}, \ldots, \mathbf{m}_1^{(n)}\right)$ and $\sum_{i=1}^{n} \mathbf{m}_0^{(i)} = \sum_{i=1}^{n} \mathbf{m}_1^{(i)}$.
The challenger chooses $b \in_R \{0,1\}$ and sends the encryption of $\mathbf{m}_b$ to the adversary $\mathcal{A}$: $\mathbf{c}_b = \left(c_b^{(1)}, c_b^{(2)}, \ldots c_b^{(n)}\right) = \left(\texttt{Enc}\left(PP, 1, K_1, id, \mathbf{m}_b^{(1)}\right), \ldots, \texttt{Enc}\left(PP, n, K_n, id, \mathbf{m}_b^{(n)}\right)\right)$

Query 2. As in the first query phase, $\mathcal{A}$ adaptively chooses and sends to the challenger a sequence of identifiers and a sequence of n-tuples of messages, and requests the encryptions. He is not allowed to choose the identifier id used in the Challenge phase.

Output. The adversary outputs a bit $b' \in \{0,1\}$.

The advantage of $\mathcal{A}$ is $\left|\Pr\left[b' = b\right] - \frac{1}{2}\right|$. A PV-PDA scheme is *secure* if it is secure against Game 1.

Game 2 (Verifiability)

Setup. The challenger obtains $(PP, EK, \{K_i\}_{i=1}^n) \leftarrow \texttt{Setup}(\lambda, n)$ and gives EK and PP to the adversary $\mathcal{A}^{verif}$.

Queries. $\mathcal{A}^{verif}$ adaptively specifies a sequence of n-tuples of messages. For $j = 1, 2, \ldots$ the j-th n-tuple is $\mathbf{m}_j = \left(\mathbf{m}_j^{(1)}, \ldots, \mathbf{m}_j^{(n)}\right)$. The challenger chooses a corresponding sequence of identifiers id_j from $\{0,1\}^\lambda$. For $j = 1, 2, \ldots$ he computes $c_j^{(i)} \leftarrow \texttt{Enc}(PP, u_i, K_i, id_j, \mathbf{m}_j^{(i)})$ for $i \in \{1, \ldots, n\}$ and gives it to $\mathcal{A}^{verif}$ (who can compute $(\mathbf{M}_j, \sigma_j) \leftarrow \texttt{Eval}(PP, EK, id_j, c_j^{(1)}, \ldots, c_j^{(n)})$ where $\mathbf{M}_j$ is the sum of the messages in the n-tuple $\mathbf{m}_j$).

Output. $\mathcal{A}^{verif}$ outputs $id' \in \{0,1\}^\lambda$, a message $\mathbf{M}'$ and a verification material σ'.
$\mathcal{A}^{verif}$ wins the game if $\texttt{Verify}(PP, \mathbf{M}', \sigma', id') = 1$ and either $id' \neq id_j$ for any j used in the query phase (type-1 forgery), or $id' = id_j$ for some j but $\mathbf{M}' \neq \mathbf{M}_j$ (type-2 forgery).

The advantage of $\mathcal{A}^{verif}$ is the probability that $\mathcal{A}^{verif}$ wins Game 2. If a PV-PDA scheme is secure against Game 2 then it is *verifiable*.

We briefly discuss how Game 1 captures the intuition on security and Game 2 on verifiability. Recall that regarding security we require (1) indistinguishability of messages, (2) uncombinability of messages from different sessions, and (3) that it is impossible to compute the sum, if not every user contributes its data.

Concerning (1), it is enough to show indistinguishability of messages from the same user in different sessions and messages from different users in the same session, because it already implies that messages from different users in different sessions are indistinguishable (as computational indistinguishability is an equivalence relation, hence transitive), and the same user in the same session never sends more than one message.

If two messages were not indistinguishable in the same session, the adversary in Game 1 would put one of them in $\mathbf{m}_0$ and the other in $\mathbf{m}_1$ thus he could guess which one was encrypted by the challenger and would win the game. If an adversary could distinguish two messages from different sessions, he would win Game 1 by querying one of them in any query phase and putting the other in, say, $\mathbf{m}_0$.

Concerning (2), if an adversary can *combine* messages from different sessions then he can easily win Game 1. He chooses $\mathbf{m}_0$ and $\mathbf{m}_1$ such that $\mathbf{m}_1^{(l)} \neq \mathbf{m}_0^{(l)}$ and combines $c_b^{(l)}$ from the challenge phase with the messages from a session in the query phase and he can see how the sum changes in that session, thus finding out the message and $\mathbf{m}_b^{(l)}$ from the challenge session.

Regarding (3), if an adversary can compute the sum of some messages without having the ciphertext for *all the messages* in that session then he can win Game 1 in the following way. If he can compute $\sum_{i \in \mathcal{I}} \mathbf{m}_b^{(i)}$ for some index set $\mathcal{I} \subsetneq \{1, \ldots, n\}$, he just have to choose $\mathbf{m}_0$ and $\mathbf{m}_1$ in such a way that $\sum_{i \in \mathcal{I}} \mathbf{m}_0^{(i)} \neq \sum_{i \in \mathcal{I}} \mathbf{m}_1^{(i)}$ and he can tell what is b by computing $\sum_{i \in \mathcal{I}} \mathbf{m}_b^{(i)}$.

Regarding verifiability, recall that we require that the SP can *only* produce a verification material for the sum of the users' values. If the verifiability property does not hold for a scheme, that means that it is possible to produce a verification material for a message different from the sum of messages, thus winning Game 2.

B. Our Scheme

In this section we describe a concrete instantiation of a PV-PDA scheme with n users. Recall that each user sends exactly one encrypted message to the SP per session. Messages are from message space $\mathcal{M}$, which has polynomial size in the security parameter λ. In the following, a message is formatted as an $N+n$ dimensional vector that contains the actual payload in the form of N values and an n-dimensional unit vector having a '1' at position i for the i-th user and '0's otherwise. The unit vector part is required to prove verifiability.

We achieve verifiability by encrypting not the plaintext values of the users but a signature of them. The signature we use is based on the network coding signature scheme of Boneh et al. [5], but we use other security assumptions and in our case, when we have the signature for some values (with the same session identifier), it is only possible to compute the signature for the sum of these values, and not for any linear combination of them (as in case of Boneh et al.). We stress that the signatures for the individual values are never in the plain in our scheme, only the signature for the sum of them is revealed.

Every session is described by a unique and unpredictable identifier that can contain some information about the scheme itself (for instance, a time stamp or the type of value) and some randomness.

In our scheme, the SP can only learn the sum of the users' values per session and nothing else. Because the SP outputs a verifiable result, the scheme works even with an untrusted server.

Furthermore, the communication between the users and the SP does not require interaction: the users can simply upload their values. However, before the first session, the initial setup is either interactive between the participants using secure multi-party computation protocols, or it is performed by a trusted Registration Authority that is no longer present after distributing the keys and parameters of the scheme. For simplicity, we use a Registration Authority in this paper.

Our PV-PDA scheme is described by the following four algorithms.

- $\texttt{Setup}(1^\lambda, n, N)$: given a security parameter λ, a number of users n, and the amount N of values that can be contained in a message, it outputs $(PP, EK, K_1, \ldots, K_n)$ with $PP = (\mathcal{G}_0, H_1, H_2, h, h^\alpha, n, g_1, \ldots, g_{N+n})$, where $\mathcal{G}_0 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ is a collection XDH group parameters, $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$ prime and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable bilinear

map. $H_1, H_2 : \{0,1\}^* \to \mathbb{G}_1$ are cryptographic hash functions, h is a generator in $\mathbb{G}_2$ and α is randomly chosen from $\mathbb{F}_p$. $EK := k$ is the evaluation key with $k := k_1 + \cdots + k_n$ for $k_i \in_R \mathbb{F}_p$ $(i = 1, \ldots, n)$ and $K_i := (k_i, \alpha) \in \mathbb{F}_p^2$ is user u_i's secret key. Finally, $g_1, \ldots, g_{N+n} \in_R \mathbb{G}_1$ are random generators of $\mathbb{G}_1$.

- $\texttt{Enc}(PP, u_i, K_i, id, \mathbf{m}^{(i)})$: given public parameters parameters PP, a user identifier u_i with corresponding secret key K_i, a session identifier $id \in \{0,1\}^*$ and a message $\mathbf{m}^{(i)} = (m_1^{(i)}, \ldots, m_{N+n}^{(i)}) \in \mathbb{F}_p^{N+n}$, where $m_{N+i}^{(i)} := 1$ and $m_{N+j}^{(i)} := 0$ $(j \neq i, j = 1, \ldots, n)$, it outputs an encryption of the message $\mathbf{m}^{(i)}$ as

$$c^{(i)} = H_1(id)^{k_i} \left(\prod_{j=1}^{n} H_2(id, j)^{m_{N+j}^{(i)}} \prod_{u=1}^{N} g_u^{m_u^{(i)}} \right)^{\alpha}.$$

- $\texttt{Eval}(PP, EK, id, c^{(1)}, \ldots, c^{(n)})$: given public parameters PP, an evaluation key EK, and ciphertexts $c^{(1)}$, $c^{(2)}$, $\ldots$, $c^{(n)}$ of messages $\mathbf{m}^{(1)}$, $\mathbf{m}^{(2)}$, $\ldots$, $\mathbf{m}^{(n)}$ from session id, it outputs $\mathbf{M} = \sum_{i=1}^{n} \mathbf{m}^{(i)}$ and a signature

$$\sigma(\mathbf{M}, id) = H_1(id)^{-k} \prod_{i=1}^{n} c^{(i)}.$$

- $\texttt{Verify}(PP, \mathbf{M}, \sigma(\mathbf{M}, id), id)$: given public parameters PP, message $\mathbf{M}$, signature $\sigma(\mathbf{M}, id)$, and session identifier id, it computes

$$\gamma_1(PP, \sigma(\mathbf{M}, id)) := e(\sigma(\mathbf{M}, id), h) \text{ and}$$

$$\gamma_2(PP, id, \mathbf{M}) = e\left(\prod_{j=1}^{n} H_2(id, j)^{M_{N+j}} \prod_{u=1}^{N} g_u^{M_u}, h^{\alpha} \right).$$

If $\gamma_1(PP, \sigma(\mathbf{M}, id)) = \gamma_2(PP, id, \mathbf{M})$ and the last n coordinates of $\mathbf{M}$ are $(1, \ldots, 1) \in \mathbb{F}_p^n$ it outputs 1 (meaning acceptance), otherwise it outputs 0 (meaning rejection).

For any $(PP, EK, K_1, \ldots, K_n)$ obtained from the $\texttt{Setup}$ algorithm and for any id, the output of $\texttt{Verify}(PP, \mathbf{M}, \sigma(\mathbf{M}, id), id)$ is 1 if and only if $(\mathbf{M}, \sigma(\mathbf{M}, id)) \leftarrow \texttt{Eval}(PP, EK, id, c^{(1)}, \ldots, c^{(n)})$.

Note that in the $\texttt{Eval}$ algorithm the SP will need to find out the message $\mathbf{M}$ from the signature $\sigma(\mathbf{M}, id)$ which is done by running the $\texttt{Verify}$ algorithm on all possible messages until it accepts the signature-message pair (exhaustive search). Therefore, the number of possible values has to be polynomial in the security parameter for the exhaustive search to run in polynomial time. We note that this is a common property of DL-based encryption schemes with additively homomorphic properties (e.g., [6]).

While the $\texttt{Enc}$ algorithm appears to perform $N + n + 2$ exponentiations and n evaluations of hash function H_2, we stress that taking advantage of the structure of $\mathbf{m}^{(i)}$ (i.e., the last n coordinates consist of $n-1$ zeros and a single one), H_2 is only called once and, for N dimensional user inputs, there are only $N+2$ exponentiations required. Thus, the extension of the messages with a unit vector does not reduce performance significantly and the number of operations required by the $\texttt{Enc}$ algorithm does not depend on the number of users.

To prove correctness of the same we have to show that for public parameters PP, evaluation key EK, session id and messages $\mathbf{m}^{(1)}, \ldots, \mathbf{m}^{(n)}$

$$\gamma_1(PP, \sigma(\mathbf{M}, id)) = \gamma_2(PP, id, \mathbf{M}).$$

The left side of this equation is $e\left(\sigma\left(\mathbf{M}, id\right), h\right)$ and the right side equals $e(\prod_{j=1}^{n} H_2(id, j)^{M_{N+j}} \prod_{u=1}^{N} g_u^{M_u}, h^{\alpha})$. Due to the bilinearity and non-degeneracy of e it suffices to show that

$$\sigma\left(\mathbf{M}, id\right) = \left(\prod_{j=1}^{n} H_2(id, j)^{M_{N+j}} \prod_{u=1}^{N} g_u^{M_u} \right)^{\alpha}.$$

By definition, the left-hand side is $H_1(id)^{-k} \prod_{i=1}^{n} c^{(i)}$, which (because of the choice of EK: $k = k_1 + \cdots + k_n$) equals to

$$\prod_{i=1}^{n} \left(\prod_{j=1}^{n} H_2(id, j)^{m_{N+j}^{(i)}} \prod_{u=1}^{N} g_u^{m_u^{(i)}} \right)^{\alpha},$$

which is the same as the right-hand side, because $M_j = m_j^{(1)} + \cdots + m_j^{(n)}$ for $j \in \{1, \ldots, N+n\}$.

C. Security of our scheme

Theorem 1: The PV-PDA scheme is secure in the random oracle model if the XDH and the strong Co-CDH assumptions hold in $\mathcal{G}_0 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$.

We prove the scheme's security under the XDH assumption in Lemma 1 by showing that if there is a successful PPT adversary against Game 1 in our scheme then there is a successful PPT adversary against the DDH problem in $\mathbb{G}_1$ (which would contradict the XDH assumption). We prove the scheme's verifiability under the XDH and the Co-CDH assumption in Lemma 2.

Lemma 1 (Security): Let $\mathcal{A}$ be a PPT adversary against Game 1 in our scheme with advantage $\frac{1}{2} + \varepsilon(\lambda)$ in the random oracle model (ε is a non-negligible function). Then there is a PPT adversary $\mathcal{B}$ against DDH in group $\mathbb{G}_1$ with advantage $\frac{1}{2} + \frac{\varepsilon(\lambda)}{2}$.

Proof: We describe how to construct an adversary $\mathcal{B}$ that solves the DDH problem interacting with $\mathcal{A}$ in Game 1. The security analysis will view hash function H_1 as a random oracle, controlled by $\mathcal{B}$.

Given a DDH instance $(\mathbb{G}_1 = \langle g \rangle, g, g^x, g^y, z)$ where $x, y \in \mathbb{Z}_p$ are chosen randomly and $z \in \mathbb{G}_1$, the challenger flips a random coin b and sets $z = g^{xy}$ if $b = 1$ and $z \in_R \mathbb{G}_1$ if $b = 0$. $\mathcal{B}$ has to decide if $z = g^{xy}$ or not, so he will output a bit b' and wins the game if $b' = b$. For simplicity we use the notation $X := g^x, Y := g^y$.

$\mathcal{B}$ first provides $\mathcal{A}$ with the public parameters $PP = (\mathcal{G}_0, H_1, H_2, h, h^{\alpha}, n, g_1, \ldots, g_{N+n})$ and the evaluation key

EK of our PV-PDA scheme. Here, $\mathcal{G}_0 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ is a bilinear group tuple where the XDH assumption holds, $H_2 : \{0,1\}^* \to \mathbb{G}_1$ is an arbitrarily chosen hash function, $h \in_R \mathbb{G}_2$, $\alpha \in_R \mathbb{F}_p$, n is the number of messages per session and $g_1, \dots g_{N+n} \in_R \mathbb{G}_1$. $\mathcal{A}$ also has access to hash function H_1, which is modelled as a random oracle. We describe later how $\mathcal{B}$ controls it.

We choose the evaluation key $EK = k \in_R \mathbb{F}_p$. The only requirement for the user secret key k_i ($i \in \{1, \dots, n\}$) is that they form a random partition of k: $k_1 + \dots + k_n = k$. k_i is never known by adversary $\mathcal{A}$, and it is only used in computing $H_1(id)^{k_i}$, and now H_1 is a random oracle controlled by $\mathcal{B}$. Therefore, as long as we make sure that $H_1(id)^{k_1} \cdots H_1(id)^{k_n} = H_1(id)^k$, we do not need to know k_i for every i. This way, all the parameters and keys meet the requirements of a PV-PDA scheme.

$\mathbf{H_1}$-queries. $\mathcal{A}$ can query H_1 at any time. In order to answer the queries consistently, $\mathcal{B}$ keeps an initially empty H^{list} of pairs $\langle id_i, r_i \rangle$, where $r_i \in \mathbb{F}_p$, and answers the queries to H_1 with identity id_i as follows:

- *First query phase of Game 1*:
  - If id_i is already in H^{list} then $\mathcal{B}$ does a table look-up in H^{list} and sends X^{r_i}.
  - If id_i is not in H^{list} yet then $\mathcal{B}$ chooses $r_i \in_R \mathbb{F}_p$ and checks if r_i is in H^{list} already. If it is, then $\mathcal{B}$ chooses another r_i randomly from $\mathbb{F}_p$ and repeats the check, otherwise he answers with X^{r_i} and adds $\langle id_i, r_i \rangle$ to H^{list}.
- *Challenge phase of Game 1*: In this phase $\mathcal{A}$ has to choose an identifier id for the challenge messages such that $id \neq id_i$ for any id_i queried earlier.
  - To the query $H_1(id)$ $\mathcal{B}$ responds with X and adds $\langle id, 1 \rangle$ to H^{list}.
  - To other queries, $\mathcal{B}$ responds the same way as in the first query phase.
- *Second query phase of Game 1*: The same as the first query phase.

$\mathcal{B}$ is now using $\mathcal{A}$ to break the DDH challenge (g, g^x, g^y, z), where $\mathcal{B}$ has to output 1 if $z = g^{xy}$. To avoid confusion, we use the notation $\overline{id}_i$ and $\overline{\mathbf{m}}_i$ for identifiers and messages in the query phase, but id and $\mathbf{m}_d$ in the challenge phase.

First query phase of Game 1. $\mathcal{A}$ sends a sequence $\overline{id}_1, \overline{id}_2, \dots$ of identifiers and a sequence $\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2, \dots$ of n-tuples of messages to $\mathcal{B}$, so $\overline{\mathbf{m}}_i$ contains all messages from all n users in session $\overline{id}_i$. To encrypt the messages, $\mathcal{B}$ picks random $T_2, \dots, T_{n-1} \in_R \mathbb{G}_1$ and sets $T_1 := z$, sets $T_n := X^k \prod_{i=1}^{n-1} T_i^{-1}$.

Therefore, we have chosen $k_1, \dots, k_n$ implicitly, such that they satisfy $k_1 + \dots + k_n = k$ and such that $T_j = H_1(id)^{k_j} = X^{k_j}$ ($j = 1, \dots, n$) for the challenge id and $T_j^{r_l} = H_1(\overline{id}_l)^{k_j} = X^{r_l k_j}$ for a query identifier $\overline{id}_l$.

Then, for $\overline{id}_l$ and $\overline{\mathbf{m}}_l = \left(\overline{\mathbf{m}}_l^{(1)}, \dots, \overline{\mathbf{m}}_l^{(n)}\right)$ (for $l \geq 1$) and for $j \in \{1, \dots, n\}$ the ciphertexts are computed as

$$c_l^{(j)} = T_j^{r_l} \left(\prod_{i=1}^{n} H_2(\overline{id}_l, i)^{m_{l,N+i}^{(j)}} \prod_{u=1}^{N} g_u^{m_{l,u}^{(j)}} \right)^{\alpha}$$

where r_l is the randomness used in the response to the H_1-query for id_l and $\overline{\mathbf{m}}_l^{(j)} = (m_{l,1}^{(j)}, \dots, m_{l,N+n}^{(j)})$.

Challenge phase of Game 1. $\mathcal{A}$ sends an identifier id such that $id \neq \overline{id}_i$ for any i, and two n-tuples of messages $\mathbf{m}_0 = \left(\mathbf{m}_0^{(1)}, \dots, \mathbf{m}_0^{(n)}\right)$ and $\mathbf{m}_1 = \left(\mathbf{m}_1^{(1)}, \dots, \mathbf{m}_1^{(n)}\right)$ such that $\sum_{i=1}^{n} \mathbf{m}_0^{(i)} = \sum_{i=1}^{n} \mathbf{m}_1^{(i)}$. $\mathcal{B}$ flips a coin $d \in_R \{0,1\}$ and encrypts $\mathbf{m}_d$ under id in the following way.

$$c_d^{(j)} = T_j \prod_{i=1}^{n} H_2(id, i)^{\alpha m_{d,N+i}^{(j)}} \prod_{u=1}^{N} g_u^{\alpha m_{d,u}^{(j)}} \text{ for } j \in \{1, \dots, n\}.$$

Second query phase of Game 1. The same as the first query phase, only this time $\mathcal{A}$ cannot query the same id queried in the challenge phase.

Output. At some point $\mathcal{A}$ outputs a bit d'.

We make the following claims:

Claim 1: With the knowledge of $\mathcal{A}$, the ciphertexts he gets in the query phase are indistinguishable from the ones that the `Enc` algorithm would output. $\mathcal{A}$ can run the `Eval` and the `Verify` algorithms on her inputs, and these will output correct values.

Claim 2: With the knowledge of $\mathcal{A}$, the ciphertexts he gets in the challenge phase are indistinguishable from the ones that the `Enc` algorithm would output. $\mathcal{A}$ can run the `Eval` and the `Verify` algorithms on her inputs, and these will output correct values.

Claim 3: If $b = 0$ (z is a random element in $\mathbb{G}_1$), the encryptions $\mathcal{A}$ gets in the challenge phase are information theoretically secure so $\Pr[d' = d | b = 0] = \frac{1}{2}$ (i.e., $\Pr[d' \neq d | b = 0] = \frac{1}{2}$). But if $b = 1$ ($z = g^{xy}$) then the messages are properly encrypted according to our PV-PDA scheme and $\mathcal{A}$ has an advantage of $\frac{1}{2} + \varepsilon(\lambda)$ to break Game 1: $\Pr[d' = d | b = 1] = \frac{1}{2} + \varepsilon(\lambda)$.

To complete the proof of Lemma 1 we only have to prove Claim 1, 2 and 3, because if $\mathcal{B}$ chooses $b' = 1$ when $d' = d$ and $b' = 0$ when $d' \neq d$ then he has a non-negligible advantage against the DDH game. This is because (by Claim 3) $\Pr[b' = b] = \Pr[d' = d | b = 1] + \Pr[d' \neq d | b = 0] \overset{\text{Claim 3}}{=} \frac{1}{2}\left(\frac{1}{2} + \varepsilon(\lambda)\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\varepsilon(\lambda)}{2}$ and it is non-negligible being $\varepsilon(\lambda)$ and thus $\frac{\varepsilon(\lambda)}{2}$ also non-negligible.

Proof of Claim 1: In the first query phase, for $l \geq 1$ and $j \in \{1, \dots, n\}$, $\mathcal{A}$ will not be able to distinguish ciphertexts

$$c_l^{(j)} = T_j^{r_l} \left(\prod_{i=1}^{n} H_2(\overline{id}_l, i)^{m_{l,N+i}^{(j)}} \prod_{u=1}^{N} g_u^{m_{l,u}^{(j)}} \right)^{\alpha}$$

from "real" cipertexts because he expects

$$H_1(\overline{id}_l)^{k_j}\left(\prod_{i=1}^{n} H_2(\overline{id}_l,i)^{m_{l,N+i}^{(j)}}\prod_{u=1}^{N} g_u^{m_{l,u}^{(j)}}\right)^{\alpha},$$

where H_1 is controlled by $\mathcal{B}$ who outputs X^{r_l} for $H_1(\overline{id}_l)$, i.e., $T_j^{r_l}=H_1(\overline{id}_l)^{k_j}$.

$\mathcal{A}$ has every parameter to run $\texttt{Eval}(PP,k,\overline{id}_l,c_l^{(1)},\ldots,c_l^{(n)})$ which will output the same $(\mathbf{M}_l,\sigma(\mathbf{M}_l,\overline{id}_l))$ from the ciphertexts $c_l^{(j)}$ (for $j\in\{1,\ldots,n\}$) received from $\mathcal{B}$ as it would compute from the ciphertexts produced by the real $\texttt{Enc}$ algorithm: it computes $\sigma(\mathbf{M}_l,\overline{id}_l)$ as $(X^{r_l})^{-k}\prod_{j=1}^{n}c_l^{(j)}$ which is indeed $\prod_{i=1}^{n}H_2(\overline{id}_l,i)^{\alpha\left(\sum_{j=1}^{n}m_{l,N+i}^{(j)}\right)}\prod_{u=1}^{N}g_u^{\alpha\left(\sum_{j=1}^{n}m_{l,u}^{(j)}\right)}$, and the sum $\mathbf{M}_l$ is computed from this, using the $\texttt{Verify}$ algorithm.

$\mathcal{A}$ can also run $\texttt{Verify}(PP,\mathbf{M}_l,\sigma(\mathbf{M}_l,\overline{id}_l),\overline{id}_l)$ which will output 1 because he gets

$$e\left(\left(\prod_{i=1}^{n}H_2\left(\overline{id}_l,i\right)^{\left(\sum_{j=1}^{n}m_{l,N+i}^{(j)}\right)}\prod_{u=1}^{N}g_u^{\sum_{j=1}^{n}m_{l,u}^{(j)}}\right)^{\alpha},h\right),$$

for $\gamma_1(PP,\sigma(\mathbf{M},id))$ and by the bilinearity of e, this is the same as $\gamma_2(PP,\overline{id}_l,\mathbf{M}_l)$. □

Proof of Claim 2: As the hash query $H_1(id)$ is answered with a random element in $\mathbb{G}_1$, and due to the choice $T_j=H_1(id)^{k_j}$, $\mathcal{A}$ is not able to distinguish ciphertexts $c^{(j)}$ sent by $\mathcal{B}$ from ciphertexts output by the $\texttt{Enc}$ algorithm with the same parameters.

$\mathcal{A}$ has every parameter to run $\texttt{Eval}(PP,k,id,c^{(1)},\ldots,c^{(n)})$, and it outputs $(\mathbf{M},\sigma(\mathbf{M},id))$ correctly ($\mathbf{M}=\sum_{j=1}^{n}\mathbf{m}_d^{(j)}$). Indeed, similarly to the query phase, $\mathcal{A}$ will compute $\sigma(\mathbf{M},id)$ as $X^{-k}\prod_{j=1}^{n}c^{(j)}$, which is $\prod_{i=1}^{n}H_2(id,i)^{\alpha\left(\sum_{j=1}^{n}m_{d,N+i}^{(j)}\right)}\prod_{u=1}^{N}g_u^{\alpha\sum_{j=1}^{n}m_{d,u}^{(j)}}$. Again, it leads to the same $\mathbf{M}$.

Adversary $\mathcal{A}$ can run $\texttt{Verify}(PP,\mathbf{M},\sigma(\mathbf{M},id),id)$ which will output 1 because $e(\sigma(\mathbf{M},id),h)$ is equal to

$$e\left(\left(\prod_{i=1}^{n}H_2\left(id,i\right)^{\left(\sum_{j=1}^{n}m_{N+i}^{(j)}\right)}\prod_{u=1}^{N}g_u^{\sum_{j=1}^{n}m_u^{(j)}}\right)^{\alpha},h\right),$$

and by the bilinearity of e, this is the same as

$$e\left(\prod_{i=1}^{n}H_2\left(id,i\right)^{\left(\sum_{j=1}^{n}m_{N+i}^{(j)}\right)}\prod_{u=1}^{N}g_u^{\sum_{j=1}^{n}m_u^{(j)}},h^{\alpha}\right).$$

□

Proof of Claim 3: When $z=g^{xy}$, from $T_1=z=X^y$ we know that $k_1=y$ and $k_1+\cdots+k_n=k$, therefore $g^yg^{k_2}\cdots g^{k_n}=g^k$ and the encryption and the evaluation work exactly the same way as in a real run of our PV-PDA scheme. In contrast, when $z=g^{xa}$ for some $a\neq y$, then $k_1=a$ and while still $k_1+\cdots+k_n=k$, the equation $g^yg^{k_2}\cdots g^{k_n}=g^k$ does not hold any more.

This concludes the proof of Claim 3 and Lemma 1. □

Lemma 2 (Verifiability): If $\mathcal{A}$ is a probabilistic, polynomial time adversary with non-negligible advantage in Game 2 against our PV-PDA scheme then there exists a PPT adversary $\mathcal{B}$ that can win the strong Co-CDH game with non-negligible probability.

Proof: We can construct an adversary $\mathcal{B}$ that has a non-negligible advantage in the strong Co-CDH game interacting with $\mathcal{A}$ in Game 2 and we look at the hash function H_2 as a random oracle controlled by $\mathcal{B}$.

Let the challenge be $f,f^{\alpha},g\in\mathbb{G}_1$, $h,h^{\alpha}\in\mathbb{G}_2$ in the bilinear group tuple $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,p,e)$.

$\mathbf{H_2}$-queries. In order to answer the queries consistently, $\mathcal{B}$ keeps an initially empty H^{list} of pairs $\langle(id_j,i),r_{ji}\rangle$ and answers the query $H_2(id_j,i)$ as follows.

- If it has already been queried, return $H_2(id_j,i)$.
- If it has not been queried yet, nor has any encryption in session id_j been, then choose $\zeta_i,\tau_i\in_R\mathbb{F}_p$ and return $g^{\zeta_i}f^{\tau_i}$.
- If encryption of any message with identifier id_j has already been queried, then return $g^{\zeta_i}f^{\tau_i}$, where ζ_i and τ_i has already been chosen during the encryption.

We now simulate Game 2.

Setup: $\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,p,e,h,h^{\alpha}$ are determined by the challenge in the strong Co-CDH, H_1 and n are chosen by $\mathcal{B}$ just as the user secret keys $k_1,\ldots,k_n,\alpha$ and the evaluation key $k=\sum_{i=1}^{n}k_i$. A random oracle controlled by $\mathcal{B}$ is used instead of H_2. $\mathcal{B}$ also chooses $s_1,t_1,\ldots,s_{N+n},t_{N+n}\in_R\mathbb{F}_p$ and sets $g_i:=g^{s_i}f^{t_i}$ for all i.

Queries: When $\mathcal{A}$ queries an encryption for $\mathbf{m}_j=(\mathbf{m}_j^{(1)},\ldots,\mathbf{m}_j^{(n)})$, where the last n coordinates of $\mathbf{m}_j^{(i)}$ form the i-th unit vector, $\mathcal{B}$ chooses an identifier $id_j\in\{0,1\}^{\lambda}$ and checks if $H_2(id_j,i)$ has already been queried for any i. If it has already been queried, $\mathcal{B}$ aborts and the simulation has failed. If not, $\mathcal{B}$ computes $\zeta_i:=-\sum_{j=1}^{N}s_jm_j^{(i)}$, chooses $\tau_i\in_R\mathbb{F}_p$ and sets $H_2(id_j,i):=g^{\zeta_i}f^{\tau_i}$ for $i=1,\ldots,n$). Then he sets $\mathbf{t}:=(t_1,\ldots,t_N,\tau_1,\ldots,\tau_n)$ and computes $c_j^{(l)}=H_1(id_j)^{k_l}(f^{\alpha})^{\mathbf{m}_j^{(l)}\mathbf{t}}$.

Output: If the simulation does not abort, $\mathcal{A}$ outputs an identifier id', a message $\mathbf{M}'$ and the verification material $\sigma(\mathbf{M}',id')$. If id' has not been queried yet, then $\mathcal{B}$ sets $H_2(id',i):=g^{\zeta_i}f^{\tau_i}$ (for $i=1,\ldots,n$), $\mathbf{s}:=(s_1,\ldots,s_N,\zeta_1,\ldots,\zeta_n)$ and $\mathbf{t}:=(t_1,\ldots,t_N,\tau_1,\ldots,\tau_n)$. If $\mathbf{M}'\mathbf{s}=0$ then $\mathcal{B}$ aborts, otherwise outputs $g':=\left(\frac{\sigma(\mathbf{M}',id')}{(f^{\alpha})^{\mathbf{t}\mathbf{M}'}}\right)^{1/(\mathbf{s}\mathbf{M}')}$.

As required, the responses to the hash queries and $g_1,\ldots,g_{N+n}$ are all random elements in $\mathbb{G}_1$. The other public parameters are also distributed identically to the public parameters produced by the real $\texttt{Verify}$ algorithm.

Now we only have to prove Claim 4, 5 and 6 to conclude the proof of Lemma 2.

Claim 4: The ciphertexts output by $\mathcal{B}$ are indistinguishable from the ones produced by $\texttt{Enc}$ and $\mathcal{A}$ can successfully run the $\texttt{Eval}$ and $\texttt{Verify}$ algorithms using the hash answers computed by $\mathcal{B}$.

Claim 5: The probability that $\mathcal{B}$ aborts is negligible.

Claim 6: If $\mathcal{A}$ wins Game 2 then $\mathcal{B}$ also wins the strong Co-CDH game.

To simplify notation, when it does not cause any confusion, we will drop the index j of $\mathbf{m}_j = (\mathbf{m}_j^{(1)}, \ldots, \mathbf{m}_j^{(n)})$ and we refer to it as $\mathbf{m} = (\mathbf{m}^{(1)}, \ldots, \mathbf{m}^{(n)})$.

Proof of Claim 4: First we show that the ciphertext $H_1(id_j)^{k_l}(f^\alpha)^{\mathbf{m}^{(l)}\mathbf{t}}$ outputted by $\mathcal{B}$ is indistinguishable from the ciphertext that `Enc` would produce, $H_1(id_j)^{k_l}\left(\prod_{i=1}^n H_2(id_j,i)^{m_{N+i}^{(l)}}\prod_{u=1}^N g_u^{m_u^{(l)}}\right)^\alpha$. This, given the H_2 hash responses and the values of g_u, equals

$$H_1(id_j)^{k_l}\left(\prod_{i=1}^n \left(g^{\zeta_i}f^{\tau_i}\right)^{m_{N+i}^{(l)}}\prod_{u=1}^N \left(g^{s_u}f^{t_u}\right)^{m_u^{(l)}}\right)^\alpha = \\ = H_1(id_j)^{k_l}\left(g^{\mathbf{sm}^{(l)}}f^{\mathbf{tm}^{(l)}}\right)^\alpha,$$

where $\mathbf{sm}^{(l)} = 0$ for $l = 1, \ldots, n$ (due to the careful choice of the ζ_is), making this the same as the output of $\mathcal{B}$.

As in Lemma 1, we only have to show that the `Eval` algorithm outputs the same $\sigma(\mathbf{M}, id_j)$ on a "real" input and on the output of $\mathcal{B}$. On a "real" input, `Eval` would output $\left(f^{\mathbf{tM}}\right)^\alpha = (f^\alpha)^{\mathbf{Mt}}$, while on the input received from $\mathcal{B}$ the evaluation algorithm will output

$$H_1(id_j)^{-k}\prod_{l=1}^n c_j^{(l)} = H_1(id_j)^{-k}\prod_{l=1}^n H_1(id_j)^{k_l}(f^\alpha)^{\mathbf{m}^{(l)}\mathbf{t}} = \\ = (f^\alpha)^{\sum_{l=1}^n \mathbf{m}^{(l)}\mathbf{t}}$$

which is $(f^\alpha)^{\mathbf{Mt}}$ according to our notation.

Now we show that the verification algorithm ran by $\mathcal{A}$ will accept the input he gets from $\mathcal{B}$ and the output of `Eval`, i.e., the equation

$$e\left(\sigma\left(\mathbf{M}, id_j\right), h\right) = e\left(\prod_{l=1}^n\prod_{i=1}^n H_2\left(id_j, i\right)^{m_{N+i}^{(l)}}\prod_{u=1}^N g_u^{m_u^{(l)}}, h^\alpha\right)$$

holds and $\mathbf{M} = (M_1, \ldots, M_N, 1, \ldots, 1)$. As we have shown, the left-hand side equals $e((f^\alpha)^{\mathbf{Mt}}, h)$ and the right-hand side is

$$e\left(\prod_{l=1}^n g^{x\left(\mathbf{sm}^{(l)}\right)}f^{\mathbf{m}^{(l)}\mathbf{t}}, h^\alpha\right),$$

which is $e(f^{\mathbf{Mt}}, h^\alpha)$ if $\mathbf{sM} = 0$. By bilinearity and non-degeneracy of e, these are the same. As the last n coordinates of $\mathbf{m}^{(i)}$ form the i-th unit vector for every $i = 1, \ldots, n$ and $\mathbf{M} = \sum_{i=1}^n \mathbf{m}^{(i)}$, the last n coordinates of $\mathbf{M}$ are all 1. □

Proof of Claim 5: The simulation aborts in two cases: if during an encryption query $\mathcal{B}$ chooses an id that he has already chosen in either a hash query or an other encryption query, or if $\mathbf{sM'} = 0$.

If the number of hash queries is q_h and the number of encryption queries is q_e then the probability of id collision failure is not larger than $\frac{q_e^2}{2^\lambda} + \frac{q_e q_h}{2^\lambda}$, which is still negligible.

If $\mathcal{A}$ outputs a type-1 forgery, then id' was not queried earlier, so $\zeta_1, \ldots, \zeta_n$ are independently uniform in $\mathbb{F}_p$, even conditioned on the view of $\mathcal{A}$. So are $s_1, \ldots, s_{N+n}$, thus $\mathbf{sM'}$ is also uniformly distributed in $\mathbb{F}_p$ (as $\mathbf{M'} \neq 0$). As a conclusion, $Pr[\mathbf{sM'} = 0] = 1/p$.

If $\mathcal{A}$ outputs a type-2 forgery (so $id' = id_j$ for some id_j used in an encryption query, but $\mathbf{M'} \neq \mathbf{M}_j$), then, conditioned on $\mathcal{A}$'s view, $\mathbf{s}$ is uniform in $\langle \mathbf{m}_j^{(1)}, \ldots, \mathbf{m}_j^{(n)}\rangle^\perp$. As the `Verify` algorithm accepts $\mathbf{M'}$, we know that the last n coordinates of it are all 1, which means that $\mathbf{M'} \notin \langle \mathbf{m}_j^{(1)}, \ldots, \mathbf{m}_j^{(n)}\rangle$, otherwise it would be equal to $\mathbf{M}_j$. Therefore $\mathbf{sM'}$ is uniformly distributed in $\mathbb{F}_p$ and equals 0 with probability $1/p$. □

Proof of Claim 6: In the proof of Claim 4 we showed that if the verification algorithm outputs 1 for the public parameters given by $\mathcal{B}$, $\mathbf{M'}, \sigma(\mathbf{M'}, id')$ and id', then $\sigma(\mathbf{M'}, id') = \prod_{l=1}^n g^{x(\mathbf{sm}^{(l)})}\left(f^{\mathbf{m}^{(l)}\mathbf{t}}\right)^\alpha = g^{x(\mathbf{sM})}(f^\alpha)^{\mathbf{Mt}}$. As a consequence, if $\mathbf{sM'} \neq 0$ then

$$g' = \left(\frac{\sigma(\mathbf{M'}, id')}{(f^\alpha)^{\mathbf{tM'}}}\right)^{1/(\mathbf{sM'})}\left(\frac{g^{x(\mathbf{sM'})}(f^\alpha)^{\mathbf{M't}}}{(f^\alpha)^{\mathbf{tM'}}}\right)^{1/(\mathbf{sM'})} = g^x.$$

This concludes the proof of Claim 4 and Lemma 2. □

III. Performance analysis

The only operation (per session) the users have to perform in our scheme is encryption of their input values. In case of one dimensional inputs (which is the most common case), the encryption requires only three exponentiations and two hash function evaluations, making it suitable for devices with relatively low computational power. The computational complexity of the verification algorithm is also low, it requires one hash function evaluation, two exponentiations and two pairings. The only computationally heavy component of our scheme is the evaluation algorithm which first combines the ciphertexts from the users to form the verification material, then runs the `Verify` algorithm a maximum of $|\mathcal{M}|$ times (trying every possible value in the message space $\mathcal{M}$), requiring in total a maximum of $1+|\mathcal{M}|$ hash evaluations, $1+2\cdot|\mathcal{M}|$ exponentiations and $2|\mathcal{M}|$ pairings. In practice, usually the SP can find the result much quicker than running the verification $|\mathcal{M}|$ times, by trying values similar to the results of previous sessions.

The higher complexity of the `Eval` algorithm does not drown the practicality of our scheme, because it is executed by the SP who usually has much higher computational capacity than the other participants.

Note that if the users' data is N dimensional (thus they encrypt $N + n$ dimensional vectors), only the number of exponentiations changes: `Enc` requires $N+2$, `Eval` requires a maximum of $1+(N+1)\cdot|\mathcal{M}|$, and `Verify` requires $N+1$ exponentiations.

The above mentioned performance indicators are summarized in Table I (for one dimensional inputs).

Operation	Algorithm		
	Enc	Eval*	Verify
Pairings	-	$\leq 2 \cdot \lvert\mathcal{M}\rvert$	2
Exponentiations	3	$\leq 1 + 2 \cdot \lvert\mathcal{M}\rvert$	2
Hash function evaluation	2	$\leq 1 + \lvert\mathcal{M}\rvert$	1

Table I: Summary of the performance indicators of the algorithms of our PV-PDA scheme, in case of one dimensional data. *The indicated number is the worst case, in most application settings it would be significantly smaller.

IV. Conclusion

We constructed a new scheme for privacy preserving aggregation of time-series data that allows for the evaluation of the sum of users' private inputs. In contrast to prior solutions, the end result is publicly verifiable without assuming a (semi-)trusted service provider, third parties, or user-interaction. Verifiability and security are proven in the random oracle model under widely used assumptions.

Verifiability of data aggregation is essential: without verifiability, the trustworthiness of the output can be questionable. We stress that being able to verify the correctness of the data aggregation is even more important when the underlying to-be-aggregated data are encrypted.

Our encryption algorithm only requires $N + 2$ exponentiations (where N is the dimension of users' data) and two hash function evaluation which is little effort at the users' side. This is important in applications such as participatory sensing and smart metering, where the user devices are resource constrained.

Limitations of our system. Our system does not support a large plaintext space or statistics other than the sum. Furthermore, it does not work in a dynamic setting (where users can easily join and leave the system) and it cannot deal with node failures (where users stop contributing). Intuitively, one might propose to use a t-out-of-n approach to solve this problem. However in a non-interactive scenario this is not possible in a PV-PDA scheme: if the service provider could compute the sum of any t users (for some $t < n$), he would be able to figure out every individual value of the users by computing the sum for all the users and for every combination of $n - 1$ users.

Colluding users can compute the sum of the rest of the users but this is unavoidable in any scheme. They also do not have any other additional power by colluding. If a user colludes with the SP, then the verification is no longer trustworthy: the SP can output any possible value (that does not contradict with $\mathcal{M}$) with a valid verification. The privacy of the users is lessened the same way as it is in case of a user-user collusion. However, we stress that any users' individual value can only be learnt if all the other users collude.

It is an interesting open challenge to construct a scheme that solves (any of) the above mentioned limitations while staying verifiable and non-interactive.

Acknowledgment. This work is supported by the THeCS project as part of the Dutch national program COMMIT/ and by the Netherlands Organisation for Scientific Research (NWO) in the context of the CRIPTIM project. We would like to thank Tim van de Kamp for the valuable feedback he gave on the draft of the paper.

References

[1] Gergely Ács and Claude Castelluccia. I have a dream! (differentially private smart metering). In Tomás Filler, Tomás Pevný, Scott Craver, and Andrew D. Ker, editors, *Information Hiding - 13th International Conference, IH 2011, Prague, Czech Republic, May 18-20, 2011, Revised Selected Papers*, volume 6958 of *Lecture Notes in Computer Science*, pages 118–132. Springer, 2011.

[2] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-resistant storage via keyword-searchable encryption. *IACR Cryptology ePrint Archive*, 2005:417, 2005.

[3] Igor Bilogrevic, Murtuza Jadliwala, Jean-Pierre Hubaux, Imad Aad, and Valtteri Niemi. Privacy-preserving activity scheduling on mobile devices. In Ravi S. Sandhu and Elisa Bertino, editors, *First ACM Conference on Data and Application Security and Privacy, CODASPY 2011, San Antonio, TX, USA, February 21-23, 2011, Proceedings*, pages 261–272. ACM, 2011.

[4] Igor Bilogrevic, Murtuza Jadliwala, Praveen Kumar, Sudeep Singh Walia, Jean-Pierre Hubaux, Imad Aad, and Valtteri Niemi. Meetings through the cloud: Privacy-preserving scheduling on mobile devices. *Journal of Systems and Software*, 84(11):1910–1927, 2011.

[5] Dan Boneh, David Mandell Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, volume 5443 of *Lecture Notes in Computer Science*, pages 68–87. Springer, 2009.

[6] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[7] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[8] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Privacy-preserving stream aggregation with fault tolerance. In Angelos D. Keromytis, editor, *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, volume 7397 of *Lecture Notes in Computer Science*, pages 200–214. Springer, 2012.

[9] Emiliano De Cristofaro and Claudio Soriente. Participatory privacy: Enabling privacy in participatory sensing. *IEEE Network*, 27(1):32–36, 2013.

[10] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[11] Marc Joye and Benoît Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, volume 7859 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2013.

[12] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In Simone Fischer-Hübner and Nicholas Hopper, editors, *Privacy Enhancing Technologies - 11th International Symposium, PETS 2011, Waterloo, ON, Canada, July 27-29, 2011. Proceedings*, volume 6794 of *Lecture Notes in Computer Science*, pages 175–191. Springer, 2011.

[13] Iraklis Leontiadis, Kaoutar Elkhiyaoui, and Refik Molva. Private and dynamic time-series data aggregation with trust relaxation. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings*, volume 8813 of *Lecture Notes in Computer Science*, pages 305–320. Springer, 2014.

[14] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In Ahmed K. Elmagarmid and Divyakant Agrawal, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 735–746. ACM, 2010.

[15] Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society, 2011.

[16] Jing Shi, Rui Zhang, Yunzhong Liu, and Yanchao Zhang. Prisense: Privacy-preserving data aggregation in people-centric urban sensing systems. In *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, pages 758–766. IEEE, 2010.